

Numerical Model Construction with Closed Observables

Felix Dietrich^{†‡§}

Gerta Köster[‡]

Hans-Joachim Bungartz[§]

October 20, 2015

Abstract

Performing analysis, optimization and control using simulations of many-particle systems is computationally demanding when no macroscopic model for the dynamics of the variables of interest is available. In case observations on the macroscopic scale can only be produced via legacy simulator code or live experiments, finding a model for these macroscopic variables is challenging.

In this paper, we employ time-lagged embedding theory to construct macroscopic numerical models from output data of a black box, such as a simulator or live experiments. Since the state space variables of the constructed, coarse model are dynamically closed and observable by an observation function, we call these variables *closed observables*. The approach is an online-offline procedure, as model construction from observation data is performed offline and the new model can then be used in an online phase, independent of the original.

We illustrate the theoretical findings with numerical models constructed from time series of a two-dimensional ordinary differential equation system, and from the density evolution of a transport-diffusion system. Applicability is demonstrated in a real-world example, where passengers leave a train and the macroscopic model for the density flow onto the platform is constructed with our approach. If only the macroscopic variables are of interest, simulation runtimes with the numerical model are three orders of magnitude lower compared to simulations with the original fine scale model. We conclude with a brief discussion of possibilities of numerical model construction in systematic upscaling, network optimization and uncertainty quantification.

1 Introduction

1.1 Multiple scale systems

Many-particle systems often exhibit dynamics on several temporal and spatial scales. These systems have many degrees of freedom but the interesting variables for applications (optimization, control, prediction and analysis) are often much fewer and change slowly with time. In this case it is desirable to find a model for these slow, coarse variables. A *model* according to our definition consists of a state space A , a dynamic $f : A \rightarrow A$, and an observer function $y : A \rightarrow \mathbb{R}^m$ with $m \in \mathbb{N}$. The states $x_k \in A$ are generated by iterative application of the dynamic, starting with a given initial state x_0 , so that

$$(1) \quad x_k = f^k(x_0).$$

The initial states usually are contained in a subset $A_0 \subset A$. Note that this definition includes continuous models with flow maps F_t , where we can choose a certain $\Delta t > 0$ to get the iterative version (Eq. 1) via $f = F_{\Delta t}$.

Consider for example a train station, where passengers get on and off trains and platforms. Such a system can be modeled at a fine temporal and spatial scale, where individuals can be distinguished and react to others in split-seconds. This accurate description might not be ideal for a safety officer controlling the number of persons on the platform. In that case, a coarse description changing every few seconds and on the whole platform hides unnecessary complexity and still provides all important information. At an even coarser scale, a building manager might want to know at which time of the day the station is most crowded.

[‡]Munich University of Applied Sciences, Lothstr. 64, 80335 Munich, Germany

[§]Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany

[†]felix.dietrich@tum.de. Questions, comments, or corrections to this document may be directed to that email address. This work was partially funded by the German Federal Ministry of Education and Research through the project MultikOSi on assistance systems for urban events – multi criteria integration for openness and safety (Grant No. 13N12824).

1.2 Extracting macroscopic dynamics

Mathematically, slow-fast systems with well-separated scales can be handled with Fenichel and perturbation theory [13], as well as homogenization and averaging techniques [14]. If the microscopic description is only available as black-box legacy code, or for systems with complex descriptions at the fine scale, these analytic techniques cannot be applied or use too many additional assumptions for the transition to the macroscopic scale. In this case, numerical analysis is a good choice. In direct simulations, many-particle systems typically need large amounts of computing resources due to the fine scales and thus again complicate analyses on the coarse scale. If the complex system cannot be simulated with all degrees of freedom, dimension and model reduction techniques are needed (both linear [19, 24] and nonlinear methods [2, 14, 17, 10, 20] are available, also see references therein). The complete fine scale can be reconstructed at all times but is not used in its entirety for simulations. The equation-free framework [16], heterogeneous multi-scale methods [11] and other multi-scale analysis techniques [10] take a different approach and are designed with the premise that only the coarse variables are interesting. They assume the existence of a coarse system and use fine scale simulations to estimate the coarse dynamics.

The main problem underlying all methods above is that the coarse model is unknown. Regarding this problem, Takens and Ruelle started to develop the theory of time-lagged embedding between 1970 and 1980 [21, 27], which enables the extraction of dynamical systems for coarse variables from scalar time series data. See [8] for the multivariate case, and [18] for extraction of input-output systems from observations. The main result of Takens and Ruelle states that observations of a system provide enough information to reconstruct the dynamics on the attractors of the system. This is possible if several future (or past) observations are used as coordinates in a high-dimensional embedding. Research following this result has focused on using the manifold in the high-dimensional space for analysis and time series prediction [22], often using eigenvalue decomposition to compute a low-dimensional parametrization. This linear decomposition is also known as Singular Value Decomposition, Principal Component Analysis, Karhunen–Loève Transform, and Proper Orthogonal Decomposition. Recently, the nonlinear parametrization of the time-lagged manifold with diffusion maps [7] has been found to be optimal, in the sense that diffusion maps recover the most stable Oseledets subspace of the dynamical system [3]. A re-parametrization with diffusion maps also separates the different time scales of the dynamic on the attractor, similar to a Fourier decomposition [3].

The analyses of the time-lagged manifold mostly focused on properties and insights into the dynamics of the original system. The possibility to construct models directly from the manifold coordinates has been largely ignored. However, Takens and Ruelle state that the new variables on the time-lagged manifold, that is, the current state of a system, contain enough information to advance in time [21, 27]. This is the basic principle of a time-autonomous model. There is no need to completely describe all states for all time; The formulation of the model, including the procedure to advance by one step (discrete or infinitesimally), is enough to *generate* the evolution over time. Hence, the problem is to find variables that contain enough information to advance the system in time without needing more information. If there are such variables, this property is known under many names, for example *closure* [16], *Markov property* or *memory-less* [12], *dependent* or *explanatory variables*. Here, we use the term *closure*.

1.3 Numerical model construction

In this paper, we show how the combination of time-lagged embedding, diffusion maps and an observer function can be used to construct coarse, time-autonomous models numerically (see Figure 1). The variables on the time-lagged manifold contain enough information to advance in time, hence the system is closed. The variables are also observable through an observer function that we create numerically, hence we call the variables *closed observables*.

The constructed numerical models are one abstraction level above the equation-free computations. The modeling process starts with the fine scale, then possibly a model reduction step to be able to run the simulator for sufficiently long time periods. After this, the equation-free framework can be employed to construct the coarse observations over an even longer time period. Finally, these observations can be used with our approach to actually construct a numerical model. It is not necessary to go through all steps to be able to construct the numerical model, any sequence of observations of a system over time can be used. The key point is that the actual modeling of the original system must only occur in the first, fine scale step. Given enough accurate observations, all models on coarser scales can then be constructed automatically.

We assume the simulation of the complex system is possible but very costly. We also assume that the interesting output *changes with time*. This dynamic aspect is a distinction from surrogate models [1], which are commonly used to construct a numerical model for the response to a certain input. Surrogate models are not trying to approximate the dynamics but the mapping from input to output of a system. This mapping might need a dynamic process internally, but the resulting surrogate model ignores the process.

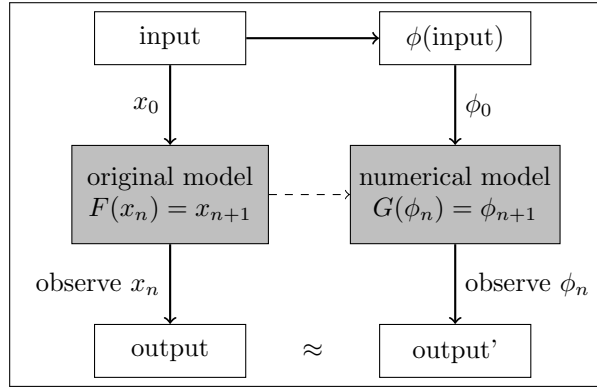


Figure 1: A given model (left) with dynamic F and current state x_n has to be transformed into a new model with dynamic G and state ϕ_n . The dimension of the new state space should be much smaller than the dimension of the original state space, but it still has to produce approximately the same output. All solid arrows depict functions, the dashed arrow indicates that the new model is created using the output of the original model.

Our paper addresses the issue of numerically constructing a model for a *dynamic* process. We state conditions when the model needs less storage than the original observations (Theorem 1). The model approximates the output of the original via an observer function. We do not modify the original model equations, but directly work with the output data. For models with differential equations, we employ the flow map with positive time delta to generate a discrete system.

The construction is different from previous approaches with diffusion maps [6], where a map back into the high-dimensional space is always necessary. In real applications, the output space is never sampled completely, and interpolation or approximation of the available values are necessary. Also, the original model must be queried many times to be able to construct a numerical model with sufficient accuracy. This is discussed in the numerical part of the paper in §3.2, together with an error analysis.

1.4 Real-world example

We use the example shown in Figure 2 to motivate the applicability of our approach. Section 5 covers this example in detail. A train enters a train station where passengers are already waiting on the platform. In this scenario, it is important to know the number of passengers over time both inside and outside the train, because very dense situations or a sudden change of density can be dangerous. We use the recently proposed Gradient Navigation Model [9] to describe and simulate the interactions between passengers and the geometry on the fine scale. Related to our general problem (Figure 1), the state space A contains elements x which hold all positions, velocities and parameters of all passengers. The function F is the simulator implementing the Gradient Navigation Model. We observe the two numbers, persons on the train and on the platform, with an observer function y on the current state x , so that $y(x) = (N_T, N_P) \in \mathbb{N}_0^2$. The passengers can be on the train or on the platform, but nowhere else. With closed observables, the resulting numerical model automatically extracts the underlying dynamics of the number of persons on both the train and the platform. In the notation from Figure 1: we construct a two-dimensional, dynamical system where

$$(2) \quad \phi_{n+1} = G(\phi_n), \quad \phi_n \in \mathbb{R}^2, \quad \phi(x_0) = \phi_0$$

and with an observer function \tilde{y} so that $\tilde{y}(\phi_n) \approx y(x_n)$. After the construction of the numerical model, we use it to analyze the chance of passengers to get out of the train within 25 and 50 seconds, given varying initial numbers of waiting passengers on the platform. The analysis, including all simulations with the constructed model, completes in seconds, whereas the same analysis with the original model would take several hours. The microscopic, individual scale and the macroscopic, platform-wide scale in this scenario differ by $\mathcal{O}(10^2)$ both temporally and spatially (see §5). This means we can use closed observables to capture a system that is about 100 times slower and larger than the microscopic system that creates the output.

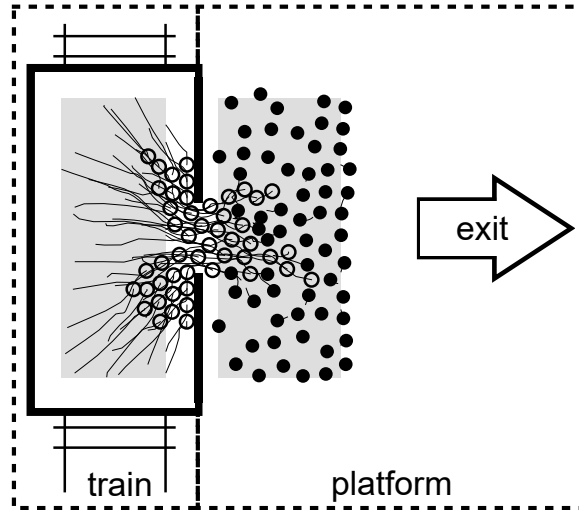


Figure 2: Setup for the real-world example. A train (left) enters the train station and passengers try to leave through the crowd of waiting passengers. Both the number of passengers inside and outside of the train are modeled using closed observables.

1.5 Outline

We introduce the mathematical framework of Takens' theorem and diffusion maps in §2. Next, numerical model construction is presented and two main theorems regarding approximation and storage efficiency are stated and proved. In §3.2, the theoretical construction of the new model is complemented with numerical algorithms and an error analysis. Two examples in §4 illustrate the ideas of the construction, the first is a two-dimensional system of ordinary differential equations and the second is a transport-diffusion system where the output is a whole function, the density over the whole domain. Section 5 describes the train station example. We conclude in §6 and add remarks on how the construction of the new model can be improved further, and how it can be used in other applications and numerical analysis.

2 Mathematical framework

The following explanations are in the context of dynamical systems theory and time series analysis. They build the theoretical basis of the construction of numerical models. Consider a time-autonomous, discrete dynamical system with state x and continuous map f of the form

$$(3) \quad x_{k+1} = f(x_k),$$

which yields the sequence of states $\{x_k\}$. Let M be the attractor of the system, that is the set of points where all states tend to as $k \rightarrow \infty$, independent of the initial state. Let the system (3) have an observer function y of the current state x_k , which for iterative application of the flow map f yields a sequence of observations $\{y_k\}$ where $y_k := y(x_k) = y(f^k x_0)$.

2.1 Time-delay embedding and Takens' theorem

The observation space consisting of all y_k does not directly represent dynamic features of the underlying system. This is because the space might not be closed dynamically: a single observation might not contain enough information to predict the next one. Takens' theorem and the resulting method of delays provide the means to construct such a dynamically closed space, which is diffeomorphic to the attractor manifold M . Let

$$(4) \quad \mathbf{h}_k = (y_k, y_{k+1}, \dots, y_{k+T-1})$$

be a vector of T observations, then for sufficiently large T Takens' theorem states the existence of a function G with

$$(5) \quad \mathbf{h}_{k+1} = G(\mathbf{h}_k).$$

Note that instead of past observations, we incorporate future observations in the current state, which will be important when constructing a numerical model later. Also see [25] for a short review on how to compute T . Takens proved the theorem with equal time sampling. See [15] for an approach where the time sampling must not be equal. In the original form of the theorem, y_k must be scalar and must also be generated without observational noise and without stochastic effects in f . The multivariate case was recently described by Deyle and Sugihara [8], and the stochastic version was described by Stark et al. [26]. The existence of the dynamic G in Eq. 5 proofs that the dynamics on attractors of the system in Eq. 3 can be reconstructed only from observations $\{y_k\}$ of the system.

2.2 Dimension reduction and observer function

The states \mathbf{h}_k are usually high-dimensional, since many y_k must be considered to close the system (T is large). Dimension reduction can be performed linearly by eigenvalue decomposition or non-linearly via kernel methods. Berry et al. showed that diffusion maps [7] are the natural choice for nonlinear dimension reduction of the time-lagged manifold, since they recover the restriction of the Lyapunov metric to the most stable Oseledec subspace [3]. Also, time-scale properties of the underlying system (Eq. 3) are separated in the reconstructed space, similar to a Fourier decomposition. The slowest dynamics can thus be separated from fast dynamics and noise, and a selection of temporal scales is possible when reconstructing the system.

Generally, the dimension reduction of the time-lagged manifold is performed via a coordinate transform from the space of the high-dimensional \mathbf{h}_k into a lower-dimensional Euclidean space. Let ϕ be the coordinate transform of \mathbf{h}_k into coordinates ϕ_k , so that $\phi_k := \phi(\mathbf{h}_k)$. Eq. 5 can now be rewritten with a dynamic \tilde{G} similar to G but acting on the transformed values ϕ_k :

$$(6) \quad \phi_{k+1} = \tilde{G}(\phi_k).$$

Eq. 6 is again a time-autonomous, discrete dynamical system. An observer function \tilde{y} can be constructed with the implicit definition

$$(7) \quad \tilde{y}(\phi_k) = \tilde{y}(\phi(\mathbf{h}_k)) = \tilde{y}(\phi(y_k, y_{k+1}, \dots, y_{k+T-1})) = y_k.$$

The observer function \tilde{y} is well defined if the coordinate transformation ϕ has an inverse ϕ^{-1} , which ensures the existence of $\tilde{y} = (y \circ \phi^{-1})_1$. Since the time-lagged manifold is diffeomorphic to the attractor manifold [3], this is always possible theoretically. In the numerical computation of the coordinate transformation with diffusion maps, this inverse is preserved for the precomputed points and can hence be also reconstructed by interpolation.

3 Numerical model construction

Based on the framework of Takens' theorem and nonlinear dimension reduction it is possible to construct numerical, macroscopic models from output data of simulations or experiments. A *numerical* model consists of all the objects of a model (state space, dynamic, observer), and all objects are created numerically via interpolation or approximation. The basic idea for the numerical models discussed here is to store the dynamics and the observer function as precomputed points and then interpolate them. In Theorem 1, we will state conditions when the construction has much lower storage requirements regarding interpolation points compared to simply storing the observed values of the original and interpolating those. The following objects have to be precomputed:

- Coordinate transformation ϕ from given input x_0 to closed observables ϕ_0
- Dynamics of closed observables, in the form $G(\phi_n) = \phi_{n+1}$ or $\Delta G(\phi_n) = \phi_{n+1} - \phi_n$
- Observer function \tilde{y} , from ϕ_n to the observation of the original observer function

With these three objects, the numerical model is independent of evaluations of the original model f (see §2). We will now show how to construct and store the interpolants for the numerical models. We also perform an error analysis of the numerical modeling process.

3.1 Storing the interpolants for the numerical model

The interpolants used by the numerical model require the storage of many data points. However, in many cases, the numerical model needs much fewer points for the same accuracy compared to the storage of all output. This is formalized in Theorem 1 and is one of the main motivations to construct the new model instead of simply storing

output and looking up the values when needed. Also, the theorem states a condition that hints towards new scientific insight: if the condition is true and the numerical model is more efficient than naive storage of observations, then the input variables have a dynamical connection, which might not be obvious and might lead to new understanding of the given process.

We formalize the storage requirements of an interpolant as follows. Consider a function $f \in C^k([0, 1]^p, \mathbb{R}^m)$ with $k \in \mathbb{N}_0$, $p, m \in \mathbb{N}$. Let $\epsilon > 0$ and consider an interpolant f_ϵ of f , so that

$$(8) \quad \|f_\epsilon - f\|_\infty < \epsilon.$$

If f is a black box (also sometimes called oracle) and can hence only be queried at distinct points $x_i \in [0, 1]^p$ to yield the values $f(x_i)$, the interpolant is constructed by sampling the space $[0, 1]^p$ with a sampling method S . This method S needs $S(\epsilon, p)$ sampling points in the construction of the interpolant f_ϵ . For the full-grid sampling method, $S(\epsilon, p)$ increases exponentially in p , since if N different samples of one coordinate axis are considered, N^p points must be stored for the p -dimensional space. If the function f has regularity $k > 0$, the error between f_ϵ and f decreases with the order $\mathcal{O}(N^{-k/p})$. More advanced sampling techniques such as sparse grids [5] exist, where the number of points for a comparable accuracy increases with $\mathcal{O}(N(\log N)^{p-1})$, mitigating the curse of dimensionality. For all sampling techniques, decreasing p without changing the smoothness properties of the function is a good way to reduce the number of points that need to be stored for f_ϵ in order to achieve an accuracy smaller than ϵ .

In the setting of this paper, we try to approximate the output of a dynamical system (Eq. 3). This means we try to find an interpolant g_ϵ for the function

$$(9) \quad g : \mathbb{N} \times A_0 \rightarrow \mathbb{R}^m, \quad g(k, x_0) = y(f^k x_0) \approx g_\epsilon(k, x_0).$$

If A_0 is p -dimensional, the function g has $p + 1$ parameters. Hence, an interpolant g_ϵ will need $S(\epsilon, p + 1)$ points for an approximation accuracy of at least ϵ . With closed observables, we only need $S(\epsilon, p)$ points for the same accuracy, which will be shown via Theorem 1.

Theorem 1. Storage reduction Define $n_0 := \dim(A_0)$, and $d := \dim(\phi(A))$. Given a sampling method S and an accuracy $\epsilon > 0$, the interpolants ϕ_I , G_I and \tilde{y}_I need at most $\mathcal{O}(S(\epsilon, n_0))$ points if the following condition is true:

$$(10) \quad d < n_0 + 1.$$

In contrast to that, storing the output of the original model needs $\mathcal{O}(S(\epsilon, n_0 + 1))$ points, where the additional dimension is approximating the time variable.

Remark 1. The dimension reduction from $p + 1$ to p variables might not seem significant. However, the reduction is for the time variable, which often is discretized much finer than the initial conditions. The gain is very large for problems which depend smoothly on the initial conditions, but need long simulation runs in time. In this case, the coordinate transformation ϕ can be interpolated with a method with low storage requirements and the time variable can be disregarded due to our numerical model construction. There are many applications where this is possible, see example 5 where the two-dimensional initial conditions are discretized with 5 and 11 points per coordinate axis, but the time is discretized with 50 points (one for each second). When using a full grid in example 4.1, the number of necessary points is reduced by 10^5 for a discretization error of $\mathcal{O}(10^{-5})$.

Theorem 1 concerns storage efficiency compared to the storage of all model output. Remarkably, the proof is independent of any special properties of closed observables, and hence the theorem applies to all models constructed by storing and interpolating data.

Proof. Theorem 1, Storage reduction Additionally, define $m := \dim(y(A))$.

1. For the n_0 -dimensional input, we need exactly d surfaces with dimension n_0 each to store the mapping from input to the new variables.
2. We also need d surfaces with dimension d each to store the dynamic G on the space D .
3. Finally, we need m surfaces with dimension d each to store the observer on the new space.
4. Since storing a finite number of surfaces does not add another dimension, the maximum dimensionality we need to store the new model is $\max(d, n_0)$.
5. To store the observed values over time, we need m surfaces (n_0 -dimensional) for each iteration step n , so in total we need $n_0 + 1$ dimensions.

6. The exact number of hyper-surfaces including dimension for the new model is given by $d \cdot \underline{d} + m \cdot \underline{d} + d \cdot \underline{n_0}$, where the underlined values stand for *dimensions* and the multiplication between a number and a dimension is noted by (\cdot) . Note that the dimension is not distributive, that is $a \cdot (\underline{b} + \underline{c}) \neq a \cdot \underline{b} + a \cdot \underline{c}$. The dimensions are additive, so $\underline{a} + \underline{b} = \underline{a + b}$. This allows for a reformulation into

$$(11) \quad \text{storage}(\text{new model}) = (d + m) \cdot \underline{d} + d \cdot \underline{n_0}$$

7. The exact number of hyper-surfaces including dimension for the storage of all observations is given by $m \cdot \underline{n_0 + 1}$. Reformulation yields

$$(12) \quad \text{storage}(\text{full output}) = m \cdot \underline{n_0 + 1}$$

8. The theorem follows from comparing and reducing the storage formulations in items 7 and 8.

□

The storage efficiency (Theorem 1) is relevant since a lot of output needs to be stored for the numerical model to be accurate. The theorem states the conditions when it is highly advantageous to construct closed observables instead of simply storing the output. The difference can be as large as the difference between the memory of a smartphone and a supercomputer (five orders of magnitude, also see example 4.1 below).

3.2 Algorithmic construction of the interpolants

In this section, we describe the algorithmic construction of the coordinate transformation, dynamics and observer function of the numerical model. A brief error analysis follows, where we verify numerically the importance of the interpolation error stated by Theorem 2.

We construct the new model with the following four steps. The detailed algorithm is shown in Figure 3.

1. Construct a set M from observations of states over time.
2. Embed the observations of M on a set H , using the current and also future observations to define coordinates of points on H .
3. Re-parametrize H with diffusion maps to form the coordinates of the closed observables.
4. Define the coordinate transformation, dynamic and the observer of the new model on the closed observables. In this step, approximation and interpolation might be necessary.

After the construction steps, the numerical model can be used for simulation and analysis, independent of the original system. A simulation with the numerical model can be performed in the following steps, given an initial state x_0 :

1. Compute initial states $\phi_0 = \phi(x_0)$ of the closed observables.
2. Iterate the given observables with the constructed dynamic, so that $\phi_{n+1} = \phi_n + \Delta G(\phi_n)$.
3. Observe the states ϕ_n with the constructed observer function \tilde{y} to generate observations of the original system:

$$(13) \quad \tilde{y}(\phi_n) \approx y(x_n) = y_n.$$

The simulation with the numerical model is independent of the original system (Eq. 3), because it does not need evaluations of the dynamic f or the observer y . The independence of the two systems enables a precomputation of the numerical model on a high performance system or with a large array of experiments, which is then condensed into the numerical model that can be used on a much less powerful device such as a smartphone.

We denote interpolated versions of the objects with a capital I as a subscript, and the error between the interpolant and the actual function by $E_{\phi, G, \tilde{y}}$, respectively, so that

- the coordinate transformation ϕ is interpolated by ϕ_I with maximum error $E_\phi := \max_{a \in A} (\phi_I(a) - \phi(a)) \in D$,
- the dynamics G is interpolated by G_I with maximum error $E_G := \max_{\phi \in D} (G_I(\phi) - G(\phi)) \in D$, and
- the observer \tilde{y} is interpolated by \tilde{y}_I with maximum error $E_{\tilde{y}} := \max_{\phi \in D} (\tilde{y}_I(\phi) - \tilde{y}(\phi)) \in \mathbb{R}^m$.

All interpolants are constructed from discrete observations and diffusion map coordinates. The numerical model with these interpolants approximates the observed values of the original model. The approximation error only depends on the interpolation errors E_ϕ , E_G and $E_{\tilde{y}}$, as stated by Theorem 2 and proofed below.

3.3 Error analysis

We now analyze the different types of errors that occur during construction and simulation of the numerical model. Theorem 2 ensures that the approximation of the original model by the numerical model only depends on the interpolation error. When generating more output with the original model, the numerical model will usually be more accurate. Note that the conditions stated in Theorem 2 are rarely all fulfilled in real applications. However, the construction process seems to be quite robust, which is mostly due to the robustness of diffusion maps regarding noise (see example 5 and the discussion in [7, 3]).

Theorem 2. Correct output of the new model Consider the state space A of the model in Eq. 3 consisting of all possible states x , and the initial states $A_0 \subseteq A$. Consider the dynamic $f : A \rightarrow A$ and the observer $y : A \rightarrow \mathbb{R}^m$. Let the objects $\phi \in C^2(A, D)$ (coordinate transformation into closed observable space D), $G \in C^2(D, D)$ (dynamic) and $\tilde{y} \in C^2(D, \mathbb{R}^m)$ (observer) be constructed numerically by the interpolation functions $\phi_I \in C^2(A, D)$, $G_I \in C^2(D, D)$ and $\tilde{y}_I \in C^2(D, \mathbb{R}^m)$. Let the interpolants have at most errors E_I , E_G and $E_{\tilde{y}}$ on the respective domains. The errors are small, so that their norms are much smaller than one. The dynamic G must have bounded, small derivatives so that for a constant $M \in (0, 1)$, $\sup_{\phi} \|DG(\phi)\| \leq M$. Then, for any given initial state $a_0 \in A_0$ and any fixed number of iterations $n \in \mathbb{N}$,

$$(14) \quad \|y(f^n a_0) - \tilde{y}_I(G_I^n \phi_I(a_0))\| \leq C_1 \left(\frac{1 - M^{n+1}}{1 - M} \right) \|E_G\| + C_2 M^n \|E_{\phi}\| + C_3 \|E_{\tilde{y}}\|.$$

The constants C_1, C_2 and C_3 are positive and independent of n and a_0 .

Remark 2. The norms of the errors of the interpolants E_{ϕ} , E_G and $E_{\tilde{y}}$ are placeholders for the actual error estimates, which differ greatly for different interpolation methods. As an example, consider a full-grid discretization with cell size h for all interpolants and orders $p_{\phi, G, \tilde{y}}$, respectively. If $M \ll 1$ and $p_{\phi} < p_G, p_{\tilde{y}}$, then Eq. 14 reduces to $\|y(f^n a_0) - \tilde{y}_I(G_I^n \phi_I(a_0))\| \leq C_4 h^{p_{\phi}}$ for a positive constant C_4 . Hence the approximation error only depends on the discretization of the coordinate transform ϕ .

Proof. Theorem 2: correct output of the new model The proof consists of the following chain of arguments, see Appendix A for details.

1. Given the exact versions of the numerical objects ϕ , G and \tilde{y} , the output is reproduced exactly. This can be proved with Takens' theorem and the assumption of a well-behaved observer y and dynamic f .
2. The interpolants of the numerical objects have small errors, so that multiplications of the errors vanish.
3. Taylor-decomposition of the interpolants (see Lemma 1 in Appendix A) yields the statement of the Theorem when combined with the first and second step of the proof.

□

The original model (or live experiment) is assumed to produce negligible errors when computing observations. Also, the matrix and eigensystem computations necessary for the diffusion map can be performed very accurately and thus their errors are neglected, too. Closed observables are based on a truncated diffusion map (see Figure 3), where the truncation is such that all information necessary to close the system is kept. Hence, we can also ignore the truncation error in this case. Three main sources of error remain:

1. Mapping from a given input to the initial state of the closed observables
2. Iterating the dynamic
3. Observing the current state

Theorem 2 states that the dynamic contributes to the total error via E_G , therefore we performed a numerical experiment (see §4.1 and Figure 8) to verify that changing the error E_G also affects the total error. The other sources of error will be treated in a future paper.

The following steps need to be completed consecutively in order to construct the numerical model:

1. Construct an approximation of the attractor manifold M (see §2):
 - Decompose the initial space A_0 into sample points $x_{1,0}, x_{2,0}, \dots$
 - Run the simulator or experiment up to time t_{end} , starting at all sampled initial points $x_{i,0}$
 - M is approximated by the observations of all states generated in the simulation or experiment.
2. Construct the time-lagged manifold H with points \mathbf{h}_k so that for all observations on M created via observing a state x_k with the observer function y ,

$$\mathbf{h}_k = (y(x_k), y(fx_k), y(f^2x_k), \dots, y(f^{T-1}x_k))$$

3. Compute pair distances d between all points $\mathbf{h}_i, \mathbf{h}_j \in H$

$$d(i, j) = \|\mathbf{h}_i - \mathbf{h}_j\|_H$$

4. Construct the diffusion map via the algorithm in [23]:
 - Transition matrix $W_{i,j} = \exp(-d(i, j)^2/\epsilon^2)$
 - Normalization matrix $N_{ii} = \sum_j W(i, j)$, so that N is a diagonal matrix containing the row sums of W
 - Solving the eigensystems $N^{-1/2}WN^{-1/2}\mathbf{v}_k = \lambda_k\mathbf{v}_k$ for all k , then sort λ_k by absolute value
 - The diffusion map coordinates are $u_k = \lambda_k N^{-1/2}\mathbf{v}_k$, so that the diffusion map is

$$\Psi : \mathbf{h}_i \mapsto (u_2, u_3, \dots).$$

The first eigenvector u_1 only contains 1 and is omitted. We also truncate the map for small λ_k and remove all u_k which only depend on previous eigenvectors. The resulting coordinates $\phi_n = \Psi(\mathbf{h}_n)$ are the closed observables.

5. The objects for the numerical model can be generated via interpolation, we use $I[\cdot]$ as a generic interpolation of the given arguments.

- The coordinate transform ϕ for the inputs $x_{i,0}$ is an interpolant of the form

$$\phi(x_{i,0}) = I[\phi_{i,0}],$$

where $\phi_{i,0} = \Psi(\mathbf{h}_{i,0}) = \Psi(y(x_{i,0}), \dots, y(f^{T-1}x_{i,0}))$.

- The dynamic ΔG is an interpolation of the difference between subsequent values of ϕ_n in time:

$$\Delta G(\phi_n) = I[\phi_{n+1} - \phi_n],$$

where the existence of ΔG is ensured by Takens' theorem (see §2). Note that an interpolation of the difference might take other values such as ϕ_{n-1} into account to be more accurate.

- The observer function \tilde{y} of the closed observables is an interpolant of the observations in the original system:

$$\tilde{y}(\phi_n) = \tilde{y}(\phi(\mathbf{h}_n)) = \tilde{y}(\phi(y_n, y_{n+1}, \dots, y_{n+T-1})) = I[y_n]$$

Figure 3: Algorithm to construct the objects of the numerical model (input mapping, dynamic and observer) from given output data.

4 Illustrating examples

We now show how numerical models can be constructed from observations of a system of ordinary differential equations, and from a transport-diffusion process. We chose the two examples to illustrate the ideas behind numerical model construction, so that the resulting numerical model can be explained from the known properties of the example. To show the potential of the approach, a more complex example of a train station is described in §5. The examples are treated similarly in the following text. First, the example is explained shortly, including the problem that arises. Second, closed observables are constructed using the algorithms described in §3.2. Third, the solution in form of a numerical model is explained and evaluated regarding efficiency and further use.

4.1 Two-dimensional initial states, one-dimensional observer

A dynamic process is often described infinitesimally via a differential equation, where the derivative of a given quantity with respect to time is set in relation to the value of the quantity itself. In this first example, we use an ordinary differential equation to describe a circling motion towards the origin (see Figure 4). The state space is two-dimensional, and the observer measures the distance of a given state to the origin. This enables the visualization of both the state space of the system and the observed values, which is not as easy if the state is very high-dimensional, such as in many-particle systems. Also, the different dimensions of the state space, the observed space and the resulting dimensions of the numerical model can be visualized easily in this low dimension. In the example presented here, closed observables provide a significant storage reduction in the sense of Theorem 1, which we demonstrate in terms of the memory capacity of a supercomputer and a smartphone.

To define the given system precisely, let $A = [-1, 1]^2$ be the state space, equal to the initial state space $A_0 = A$, $y(a) = \|a\|$ be the observer and F the push-forward by $\Delta t = 0.1$ of the system of ordinary differential equations

$$(15) \quad \frac{da}{dt} = -a + Ra, \quad a \in A,$$

where R is the rotation matrix $[0, -1; 1, 0]$. The phase diagram in this case is depicted in Figure 4. To generate

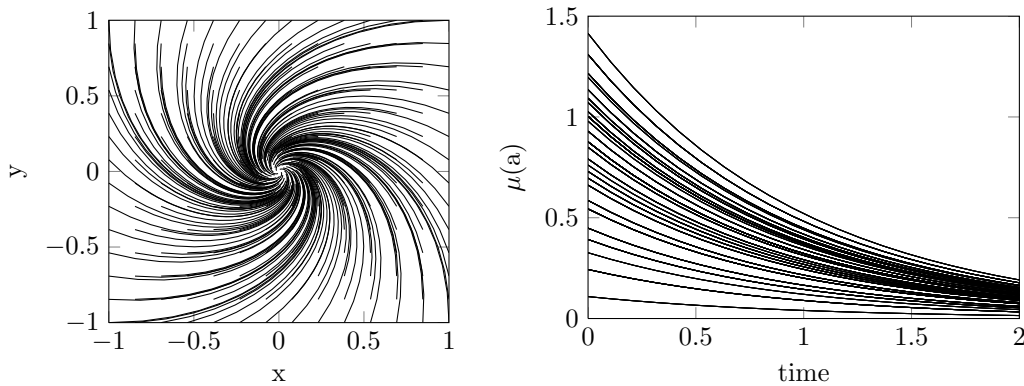


Figure 4: System with two-dimensional initial states. Starting from any point in $\mathcal{A}_0 = [-1, 1]^2$, the system spirals inward (left figure). When measured, every trajectory shows the same characteristic (right figure): the distance to the center is dropping to zero exponentially fast.

observations of the system, we initialize it at equally spaced points on A_0 and store all $y(a_n) = \|a_n\|$ of the discrete system $a_{n+1} = F(a_n)$ up to time $t_{end} = 2$. The closed observables are constructed with the algorithm described in Figure 3 with $T = 5$, which corresponds to $\Delta t = 0.5$.

Fig. 18 shows the relation between the first and following diffusion map coordinates. All but the first can be discarded, as they are functions of the first, which means that only one dimension is necessary to describe the dynamics of the system. The state in the new coordinates also moves to a steady state exponentially fast (Fig. 5). Figure 6 shows how the initial states of the original system translate to the initial states of the reduced system. In contrast to the distance to the center before (Figure 4, right), they can not directly be interpreted but must be observed first. The observer function is depicted in Figure 6 on the right.

The reduction in this example is significant. Naively storing all observations over time for all initial states would take a three-dimensional hyper-surface (see Theorem 1 and Figure 7). The reduced model with closed observables only needs two 1D-lines for dynamic and observer (Figures 5, left and 6, right), and one 2D-surface for the translation

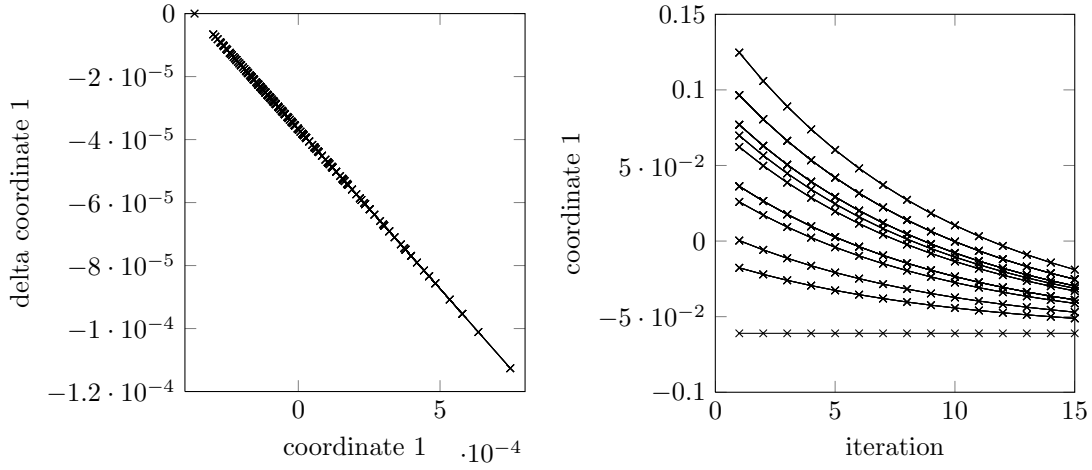


Figure 5: Left figure: dynamic on the closed observable. Right figure: trajectories in the reduced system for several initial states. The exponential decay is clearly visible, though the actual values of the coordinate have no direct meaning (e.g. are not directly the distance to the center).

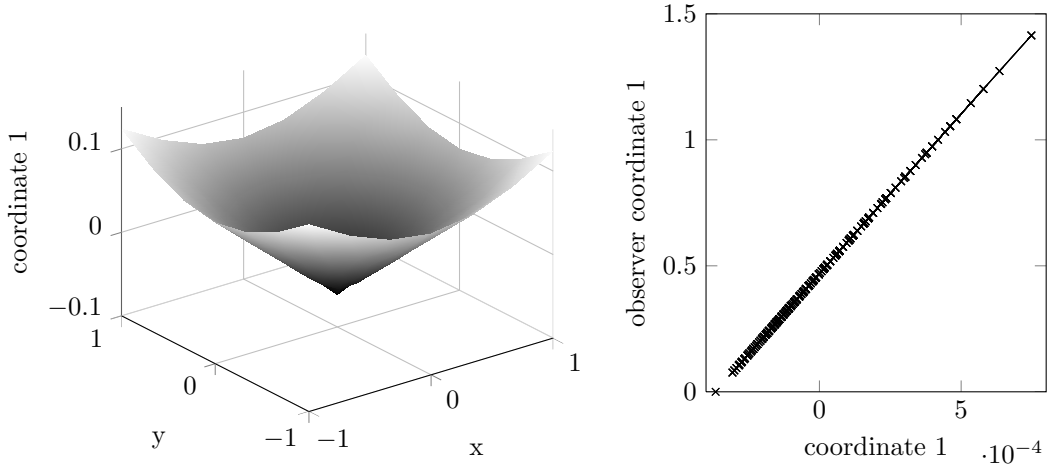


Figure 6: Left: Initial states in reduced system coordinates relative to the original coordinates (x, y) . The reduced coordinate values are symmetrical around $(0, 0)$, which is due to the choice of the observer in the original system (distance to the center). Right: observer of the closed observable ϕ_1 .

from initial state to the new coordinate (see Figure 6). Even in this low-dimensional case, the naive storage of all observations over time for all initial states would take up all memory of a supercomputer (Tianhe-2, state 2015). If the same naive sampling is applied together with closed observables, the data fits in the memory of a smartphone (Figure 7).

The success in this case can be explained by reformulating the system in Eq. 15 with the observer $y(a) = \|a\|$. This yields the new system

$$(16) \quad \frac{db}{dt} = -b,$$

where $b \in \mathbb{R}$ and $y(b) = b$. The translation from a value a to b is given by $b = \|a\| = y(a)$. Remarkably, the closed observables capture a similar system numerically.

We compared the numerical errors of experiment 4.1 by changing the number of points on the trajectories used to construct the dynamic ΔG of the numerical model. We used two numerical methods to approximate the difference $\phi_{n+1} - \phi_n$: a one-sided gradient, so that $\Delta G(\phi_n) = \phi_{n+1} - \phi_n$ and a gradient computed with central differences, so that $\Delta G(\phi_n) = (\phi_{n+1} - \phi_{n-1})/2$. The results are shown in Figure 8, where the relative error between the observations of the original and the numerical model is displayed for different numbers of numerical steps between $t = 0$ and $t = 2$. The change of the interpolation error as well as the change of the interpolation order directly

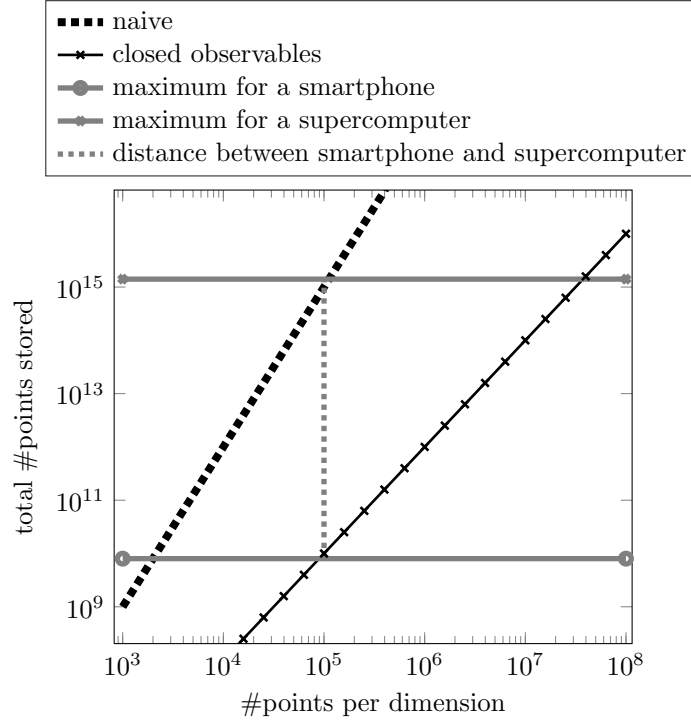


Figure 7: Storage of a model with closed observables compared to naive storage for example 2 in §4.1, with parameters $D = 1$, $M = 1$, $N_0 = 2$. In both cases we assume equal spacing of points. We assume a memory capacity of 8GB for the smartphone and $88\text{GB} \cdot 16000$ for the supercomputer (Tianhe-2, 2015).

influences the error of the numerical model, as stated by Theorem 2. In our case, the one-sided gradient has order $\mathcal{O}(\Delta t)$, and the two-sided gradient has order $\mathcal{O}(\Delta t^2)$.

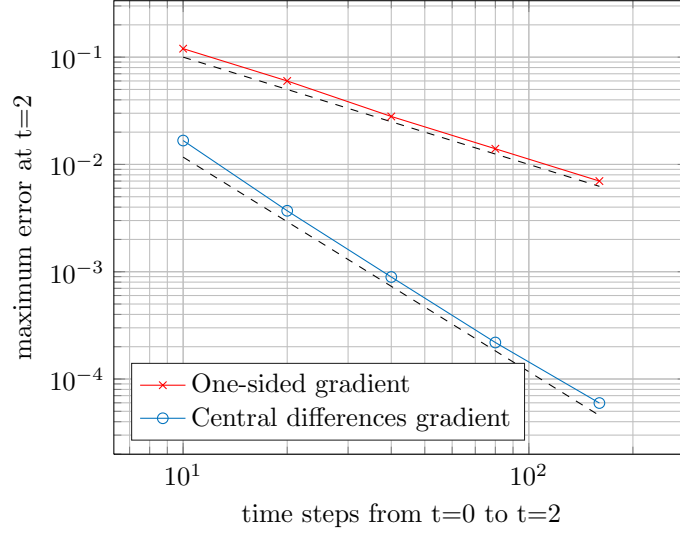


Figure 8: Maximum of relative approximation error $e = \max_t \left\| \frac{y(f^t(a_0)) - \tilde{y}(G^t(\phi(a_0)))}{y(f^t(a_0))} \right\|$ from $t = 0$ to $t = 2$ between the original system in example 4.1 and the model constructed with closed observables. The maximum is also computed over 1000 randomly chosen initial values in $A_0 = \{(0, y) | 1 \leq y \leq 3\}$. Different numerical approximations were used for ΔG : the error decreases with order $\mathcal{O}(\Delta t)$ with a one-sided gradient and $\mathcal{O}(\Delta t^2)$ for a gradient computed with central differences (see dashed lines).

4.2 Two-dimensional initial states, infinite-dimensional observer

Diffusion, transport and advection of physical quantities such as mass, concentration or energy are very common phenomena in nature. In this example, we use closed observables to numerically capture the dynamics of such processes. We provide observations of a generic quantity u over a one-dimensional domain, and change two parameters of the system to generate different results. Thus, the system as seen by the algorithms is a black box F with two parameters c and d , producing observations $u(x, t)$ for a given time t and space x (see Figure 9 and 17 in the appendix):

$$(17) \quad F_{c,d}(x, t) = u(x, t).$$

To be able to verify the results of the closed observables, we choose the classical transport-diffusion PDE in one dimension on $x \in [-0.5, 0.5]$ with periodic boundary conditions and constants for diffusion c and advection d . In our example, the transport coefficient a depends linearly on d via $a(d) = 10d$:

$$(18) \quad cu_{xx} = du_t + a(d)u_x$$

$$(19) \quad u(x, 0) = \exp(-25x^2).$$

This system is chosen because for $d \neq 0$ it can be reformulated as depending only on one coefficient:

$$(20) \quad \frac{c}{d}u_{xx} = u_t + 10u_x.$$

The reformulation shows that the underlying system is two-dimensional, because of essentially one parameter $\frac{c}{d}$ and the time dimension. Via closed observables, we are able to capture this property of the system automatically: no reformulation is necessary, we simply provide the observed values of $u(x, t)$ from the simulator. The state at time t is the function $u(\cdot, t)$, and the observer y is the identity, so that $y(u) = u$.

Even though the observed values are functions, numerical model construction creates a finite-dimensional dynamical system and the observer of the closed observables yields a function. The constructed numerical model is capable of producing different dynamics for given constants c and d , where $a(d) = 10d$. The numerical model needs two dimensions to capture the two dimensional initial conditions for diffusion and advection (see Figure 10), and three dimensions for the observer (for every pair (ϕ_1, ϕ_2) , $u(x)$ has to be stored for all x , see Figure 11). This is one dimension less than naively storing all output for all combinations of the input parameters, since for each pair (c, d) one would need to store $u(x, t)$ for all x, t , requiring storage of four dimensions.

Figure 12 shows a comparison of the error between the result of closed observables at intermediate values of c and d not used in the generation of the model and interpolation of the observations closest to the given values of c and d .

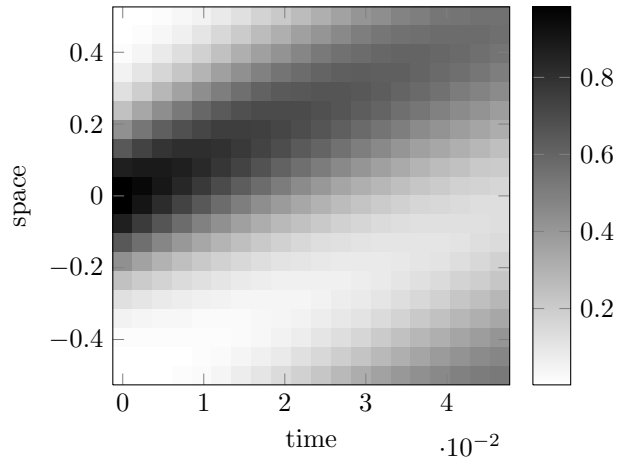


Figure 9: Solution $u(t, x)$ of Eq. 18 for diffusion $c = 3$, advection $d = 6$, and transport $a = 10 \cdot d = 60$.

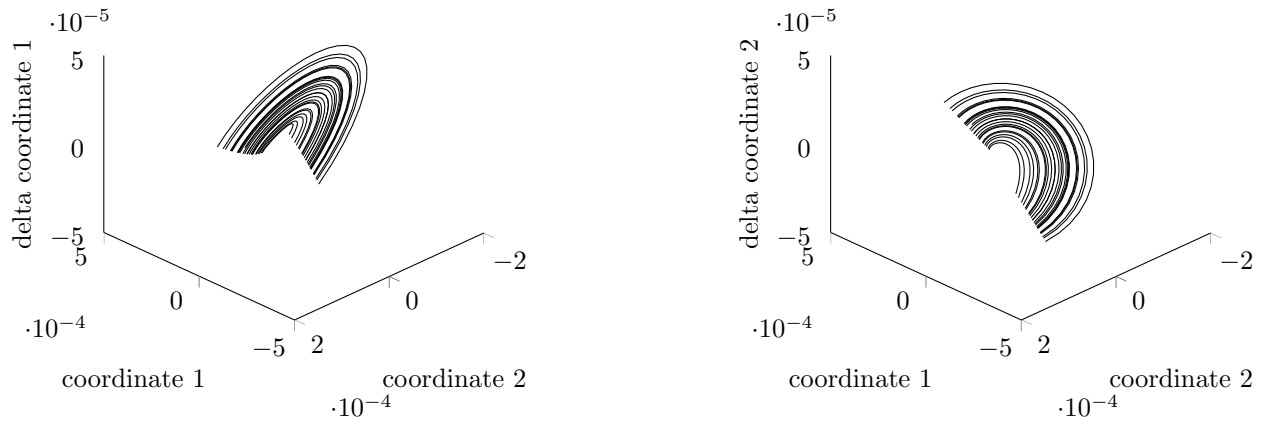


Figure 10: Dynamics of the closed observables, ϕ_1 and ϕ_2 .

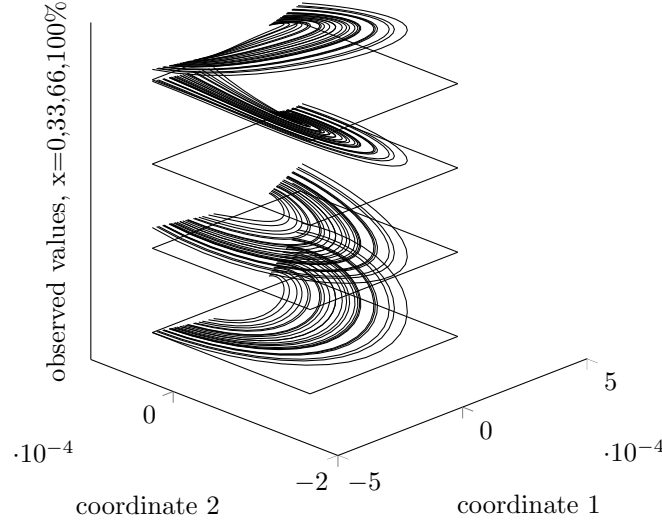


Figure 11: Slices at four points x_1, x_2, x_3, x_4 in $[-0.5, 0.5]$ of the observer of the closed observables, $\tilde{y}(\phi_1, \phi_2)(x)$. The z-value of the upper slices is translated upward, otherwise they would all be overlapping.

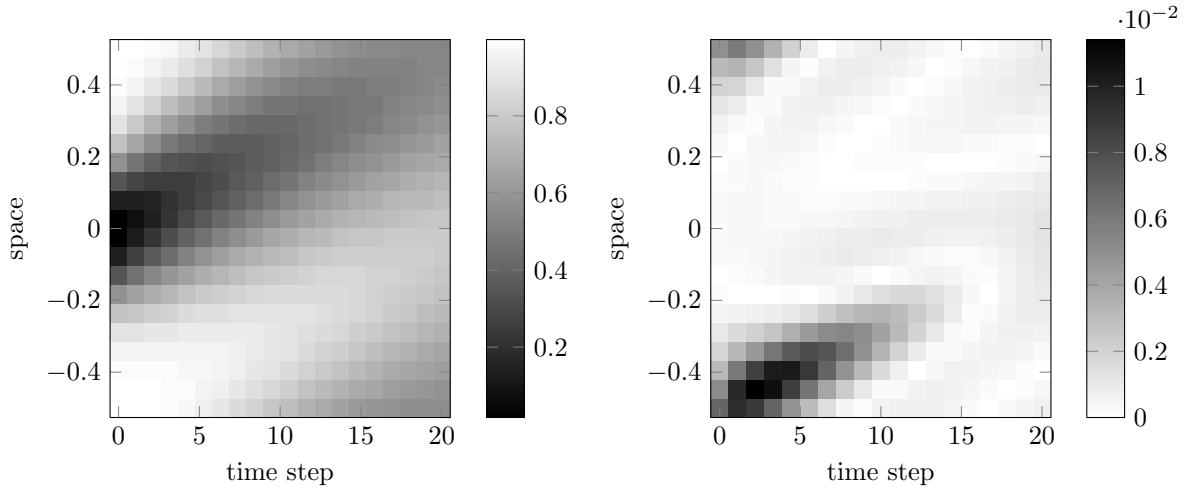


Figure 12: Left: predicted dynamics when interpolating at $c = 1.6$, $d = 2$. Right: relative error $\epsilon = \frac{|O-P|}{|O|}$ between observations O (generated with $c = 1.6$, $d = 2$ and not used in the construction of the model) and prediction P .

5 A real-world example

This section shows how to apply closed observables in a real-world example as introduced at the beginning in §1.4. Safety at train stations can be lowered dramatically when a train with many passengers arrives on an already crowded platform. In this scenario, opening the doors of the train is dangerous, as the number of persons on the platform can reach critical levels and people might get pushed on tracks or suffocate due to high interpersonal pressure. For safety personnel, the evolution of the number of persons on the train and on the platform can be crucial to decide whether to open the doors or evacuate the platform. A model of this process can provide hints on the future evolution of passenger numbers.

Consider such a train station with one track and a platform (see Figure 13). Passengers can leave the train through a door and then have to pass through waiting passengers on the platform to leave the station. There are several possibilities to simulate this system at the macroscopic scale. One could use analytical techniques such as homogenization and averaging to scale up the differential equations describing individual behaviour. This would yield a macroscopic model, possibly also a differential equation, for the exit process out of the train. Another approach is taken in the natural sciences. The microscopic simulator or a live experiment generates output data (number of persons in and out of the train at each second), starting with several different initial settings. A macroscopic model is then proposed and validated using the output. Classical surrogate models would try to approximate the output either for each time step individually, or as an input-response system. For example, such a system could yield the mean evacuation time given the initial number of pedestrians inside and out of the train. For dynamic properties of the system, such as the outflow over time, an interpolation of all output would be necessary.

Both analytic and experimental approaches yield a macroscopic model, in most cases a system of equations. The approach of a surrogate model with closed observables yields a dynamical system that can be used in simulations. However, similar to classical surrogate models it is not present as equations, but numerical data and interpolating functions. This data is generated automatically by the process described in §3.2. The result is either more storage efficient than classical surrogate models (see Thm. 1), or more accurate while using the same amount of storage.

Here, we describe how to construct the macroscopic model with closed observables. The following assumptions are used for the train setting:

Assumption 1 Passengers follow the rules of the Gradient Navigation Model [9].

Assumption 2 Desired speeds approximately obey a normal distribution with mean $1.34m/s$ and standard deviation $0.26m/s$ (experimental values from [28]).

Assumption 3 Initially, the positions of passengers in the train and on the platform are uniformly distributed. We use a five second starting phase for the distribution to settle in a state where all passengers on the platform assume their desired distances to others, and passengers in the train queue in front of the door.

Assumption 4 Waiting passengers do not strongly react to the leaving passengers, but move away a little if distances are too small. By this assumption, we exclude psychological effects such as the formation of a passage way in front of the door.

Assumption 5 Small differences in the initial positions of the passengers do not cause large differences in behavior on the system level. This allows us to start several simulation runs with the same initial numbers of passengers but small changes in positions on train and platform, and then average over the results.

We are interested in the number of passengers over time, both on the train and on the platform. A simulation with 42 passengers on the train and 48 passengers waiting on the platform results in the change of passengers depicted in Figure 13. After 15 seconds, all 90 passengers are on the platform.

The scales in this scenario differ from $\mathcal{O}(10^{-1}s)$ to $\mathcal{O}(10^1s)$ on the temporal scale (reaction time of pedestrians is set to 0.5 seconds, the analysis of the chance to get off the train is between 25 and 50 seconds), and from $\mathcal{O}(10^{-1}m)$ to $\mathcal{O}(10^1m)$ on the spatial scale (diameter of a pedestrian is $0.4m$, the platform is about $20m$ wide). This means that we use closed observables to capture a system that is $\mathcal{O}(10^2)$ slower and larger than the microscopic system that creates the output.

We denote the numbers of passenger on the train N_T and on the platform N_P . In our example, passengers cannot leave the station. Also, passengers leaving the train arrive on the platform. These facts can be combined to form the continuity equation, which in our case is present in the discrete version. Consider the outflow F_T of passengers from the train, then:

$$(21) \quad N_P(t+1) - N_P(t) = F_T(t).$$

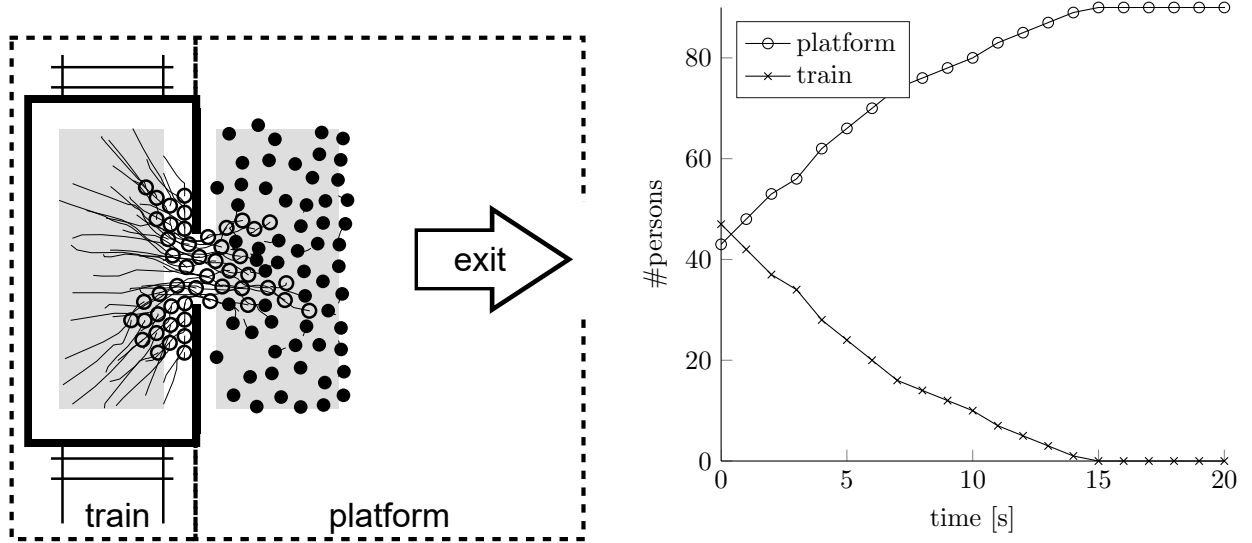


Figure 13: Left: The train station with a train and the platform with waiting passengers. Persons stepping off the train (empty circles) have to leave to the right. We are interested in the numbers of persons over time, both on the train and on the platform. Right: Evolution of the passenger number in the train and on the platform. 48 persons start in the train, 42 on the platform. After 15 seconds, all 90 passengers are on the platform.

The function $F_T(t)$ depends on interactions between passengers and the geometry and hence is very difficult to derive analytically. A typical solution in this case is to run simulations and estimate $F_T(t)$ for different numbers of passengers, which would need $\mathcal{O}(N^3)$ points for a sampling of the two parameters and time. With closed observables, the same accuracy can be achieved by storing $\mathcal{O}(N^2)$ points (see below).

To model N_P and N_T with closed observables, Eq. 21 and the consideration of the related facts are not necessary. We only need the output of the simulations started for different numbers of pedestrians on the train and on the platform. As we draw the initial positions of the persons from a uniform distribution, the output is stochastic and we have to run several simulations for the same initial numbers $N_P(0), N_T(0)$ to get a good average evolution $\bar{N}_P(t), \bar{N}_T(t)$. Table 1 shows the parameters of the simulation runs. The output $\bar{N}_P(t), \bar{N}_T(t)$ is used to construct

Parameter	Values
Simulated time on the train station	50 seconds
$N_T(0)$: Initial numbers of passengers on the train	10,20,30,40,50
$N_P(0)$: Initial numbers of passengers on the platform	0,20,40,60,80,100,120,140,160,180,200
Runs per initial value pair (N_P, N_T)	10
T: number of observations to combine for numerical model	15

Table 1: Parameters for the simulations and numerical model construction. In total, $5 \cdot 11 \cdot 10 = 550$ simulations were started. The parameter T shows how many observations are combined to form a time-lagged variable (closed observables). In the example, observations were one second apart, so that 15 second intervals are combined.

the diffusion map space D . For notational convenience, we drop the braces denoting the mean from now on. The functions ϕ , G and $\tilde{\mu}$ are created using linear interpolation by MATLABs `scatteredInterpolant`. We use the algorithm outlined in Figure 3. Figure 14 depicts the interpolating surfaces for ΔG . We need two since we employ two-dimensional closed observables, that means in this real world example,

$$(22) \quad (\phi_1, \phi_2)_{n+1} = (\phi_1, \phi_2)_n + \Delta G((\phi_1, \phi_2)_n).$$

Since we use interpolation, the original output can be recreated up to high accuracy (see Figure 19). For input not covered in the simulations, the trajectories generated by the numerical model closely predict the system behavior. Figure 15 shows a comparison between a predicted trajectory and three corresponding simulations not used for model construction ($N_P(0) = 60, N_T(0) = 35$). Note that the prediction is for the mean of the simulation runs.

An analysis of the number of passengers getting off the train in the simulated 25 or 50 seconds yields the result

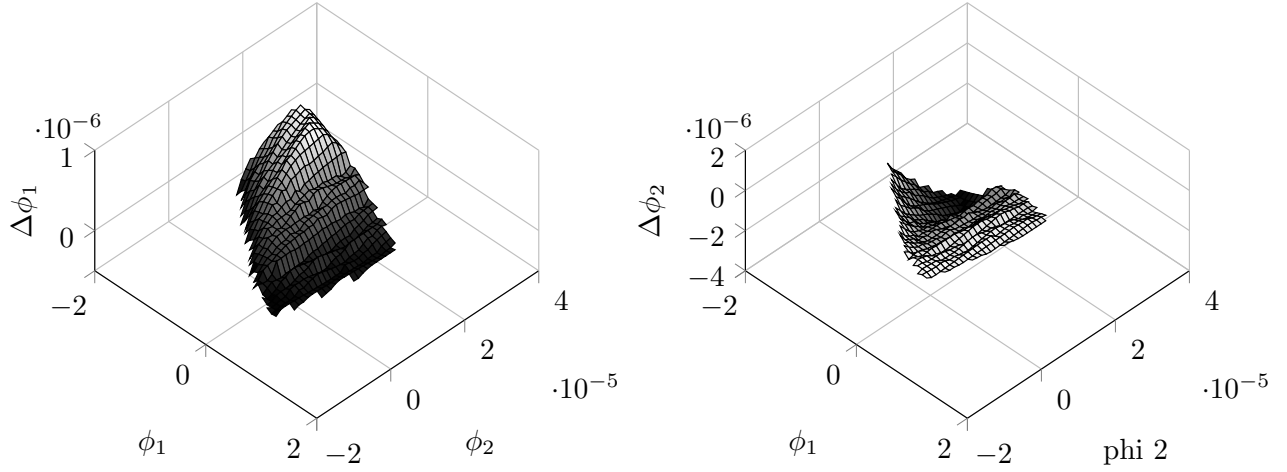


Figure 14: Evolution function $\Delta G(\phi_1, \phi_2) = (\Delta\phi_1, \Delta\phi_2)$ in the form of two functions $\Delta\phi_1 = G - \phi_1$, $\Delta\phi_2 = G - \phi_2$ on closed observables ϕ_1 and ϕ_2 .

shown in Figure 16. We compute the mean chance c for one pedestrian to get off the train in n seconds by

$$(23) \quad c(n) = 1 - \frac{\min_{t \leq n} N_T(t)}{\max_{t \leq n} N_T(t)} = 1 - \frac{\min_{t \leq n} N_T(t)}{N_T(0)}.$$

The chance to get off the train in time decreases with the number of passengers waiting on the platform. Running the necessary 400 simulations took about two seconds, whereas the generation of the data with the original model would last for hours. It would also need 10 simulation runs per data point to generate the average value, resulting in 4000 simulations for the same result.

The real-world example shows where closed observables are advantageous. Compared to the original model, creating a model of the process of stepping of the train is much more efficient computationally than the original, for both analysis and prediction (see Figure 16). Compared to storage of the full simulation output, it is also much more efficient. Theorem 1 states that in this example the constructed model is exponentially more efficient regarding data storage, as the input space is two-dimensional ($\dim A_0 = 2$) and the closed observables also have two dimensions ($\dim D = 2$). A response surface common in surrogate modeling cannot yield the same result, because it would be constructed only for certain end times (for example 25 or 50 seconds). The model constructed with closed observables can produce results up to any given end time.

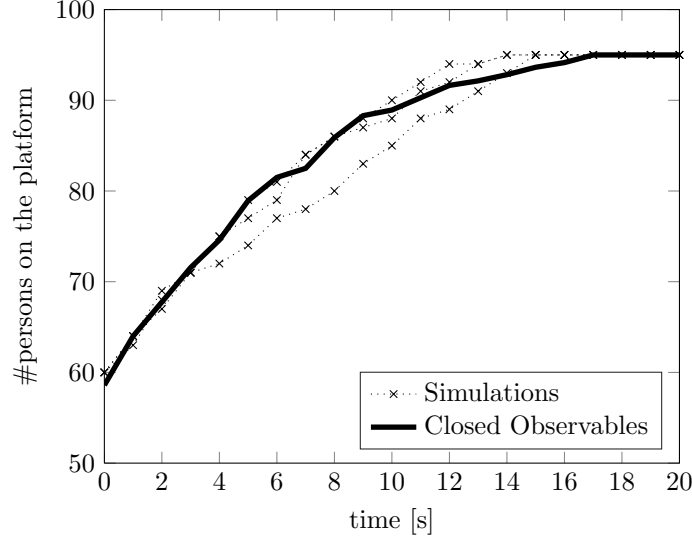


Figure 15: A prediction (bold line) of the average number of persons on the platform by the constructed model, compared to three simulation runs (dotted). The incorrect starting values for the prediction results from interpolation errors: the given initial values $(N_P(0), N_T(0)) = (60, 35)$ are transformed by ϕ into the space D and after the simulation recreated by observation with \hat{y} .

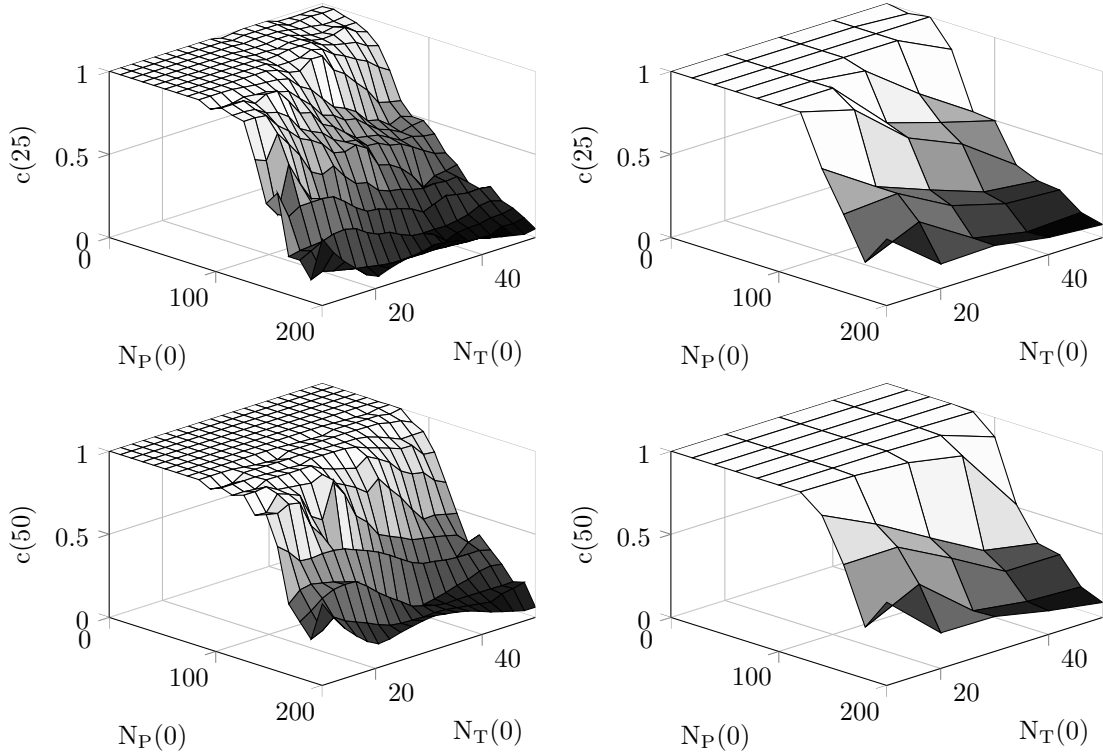


Figure 16: Left column: Prediction and analysis with the numerical model constructed with closed observables. The chance of passengers getting off the train in 25 seconds (top row) and 50 seconds (bottom row) decreases with the number of passengers waiting on the platform. Right column: The same analysis performed on the output used to construct the numerical model. Small features in the surface are not visible.

6 Conclusion

Many systems often show interesting dynamics on multiple temporal and spatial scales. If such a system is present on a fine scale, generating a model for the same process but a coarser time scale is difficult but necessary to perform analysis, optimization and control. We introduced the concept of numerical models with closed observables on a coarse scale and showed how observations of a system can be used to construct it.

In case the number of dynamically meaningful variables, called closed observables, is not higher than the number of input parameters, constructing the numerical model is more efficient than storing and interpolating the output (Theorem 1). This can easily make the difference between the memory capacity of a smartphone and a supercomputer (see example 4.1).

Two academic examples illustrated the features of our approach, from initial value mapping to storage reduction and finite system construction of PDEs. A real-world example showed that numerical model construction can extract the dynamics of macroscopic systems from fine scale simulations, with performance gains of four orders of magnitude.

Future work will focus on two major areas: a search for analytic forms of closed observables, and scaling up stochastic, fine scale systems. Numerical models can be used in various fields of numerical analysis. In network optimization, fine scale simulations could now be used to build models on the vertices or edges of the network. Systematic upscaling [4] needs models on several scales, which could be simplified with the numerical model construction ideas presented here. Uncertainty quantification can now be split into a construction phase, where the numerical model on the scale of interest is constructed with the fine scale simulator, and an analysis phase, where the high performance of the numerical model can be used to perform quantification of uncertainties without running the original simulator.

When enough experimental data is available, one could also construct numerical models directly from the observations and in this way extract macroscopic dynamics automatically from experiments.

A Lemmas, Proofs

Lemma 1. *Consider two vector spaces A and D . Let $G \in C^2(D, D)$ and $\phi \in C^2(A, D)$. Consider the interpolants of these functions, $G_I \in C^2(D, D)$ and $\phi_I \in C^2(A, D)$, with interpolation errors $E_G(\phi) := G_I(\phi) - G(\phi)$ and $E_\phi(a) := \phi_I(a) - \phi(a)$. We denote $E_G := \max_{\phi \in D} E_G(\phi)$ and $E_\phi := \max_{a \in A} E_\phi(a)$. Also, let $\|DG\|$ be bounded by $M \in (0, 1)$, so that $M \geq \sup_{\phi} \|DG(\phi)\|$. Then, for a given $a \in A$ and a fixed $n \in \mathbb{N}$,*

$$(24) \quad \|G^n(\phi(a)) - G_I^n(\phi_I(a))\| \leq C_1 \left(\frac{1 - M^{n+1}}{1 - M} \right) \|E_G\| + C_2(M^n \|E_\phi\|)$$

where the constants C_1, C_2 are positive and do not depend on a and n .

Proof. The Taylor-decomposition of a vector-valued C^2 function f at a point $x + \epsilon$ with small $\|\epsilon\|$ is given by

$$(25) \quad f(x + \epsilon) = f(x) + DF(x) \cdot \epsilon + \mathcal{O}(\|\epsilon\|^2).$$

This approximation is the basis for the following arguments. For any $\phi \in D$ and $n \geq 1$,

$$\begin{aligned} \|G_I^n(\phi)\| &= \|G_I^{n-1} \circ [G(\phi) + E_G(\phi)]\| \\ &= \|G_I^{n-2} \circ [G^2(\phi) + DG(G(\phi)) \cdot E_G(\phi) + \mathcal{O}(\|E_G(\phi)\|^2) + E_G(\phi)]\| \\ &= \|G_I^{n-2} \circ [G^2(\phi) + \mathcal{O}((M+1)\|E_G(\phi)\|) + \mathcal{O}(\|E_G(\phi)\|^2)]\| \\ &= \|G_I^{n-3} \circ [G^3(\phi) + \mathcal{O}((M(M+1)+1)\|E_G(\phi)\|) + \mathcal{O}(\|E_G(\phi)\|^2)]\| \\ &= \dots \\ &= \|G^n(\phi) + \mathcal{O}\left(\sum_{i=0}^n M^i \|E_G(\phi)\|\right) + \mathcal{O}(\|E_G(\phi)\|^2)\|. \end{aligned}$$

With $M \in (0, 1)$, we thus have

$$(26) \quad \|G_I^n(\phi)\| \leq \|G^n(\phi) + \mathcal{O}\left(\frac{1 - M^{n+1}}{1 - M} \|E_G(\phi)\|\right)\|$$

Similarly, for any $a \in A$,

$$\begin{aligned}
\|G^n(\phi_I(a))\| &= \|[G^{n-1} \circ G](\phi(a) + E_\phi(a))\| \\
&= \|G^{n-1} \circ [G(\phi(a)) + DG(\phi(a)) \cdot E_\phi(a) + \mathcal{O}(\|E_\phi(a)\|^2)]\| \\
&= \|G^{n-2} \circ [G^2(\phi(a)) + DG(\phi(a)) \cdot \mathcal{O}(M\|E_\phi(a)\|) + \mathcal{O}(\|E_\phi(a)\|^2)]\| \\
&= \dots \\
&\leq \|G^n(\phi(a))\| + \mathcal{O}(M^n\|E_\phi\|).
\end{aligned}$$

Note that for $M \in (0, 1)$, the right side converges to $\|G^n(\phi(a))\|$ if $n \rightarrow \infty$. Combining the two approximations, we conclude that for any $a \in A$ and $M \in (0, 1)$,

$$\begin{aligned}
\|G_I^n(\phi_I(a))\| &\leq \|G^n(\phi(a))\| + \mathcal{O}\left(\frac{1 - M^{n+1}}{1 - M}\|E_G\|\right) + \mathcal{O}(M^n\|E_\phi\|) \\
&= \|G^n(\phi(a))\| + C_1\left(\frac{1 - M^{n+1}}{1 - M}\right)\|E_G\| + C_2M^n\|E_\phi\|
\end{aligned}$$

for constants $C_1, C_2 > 0$ independent of a and n . □

B Figures

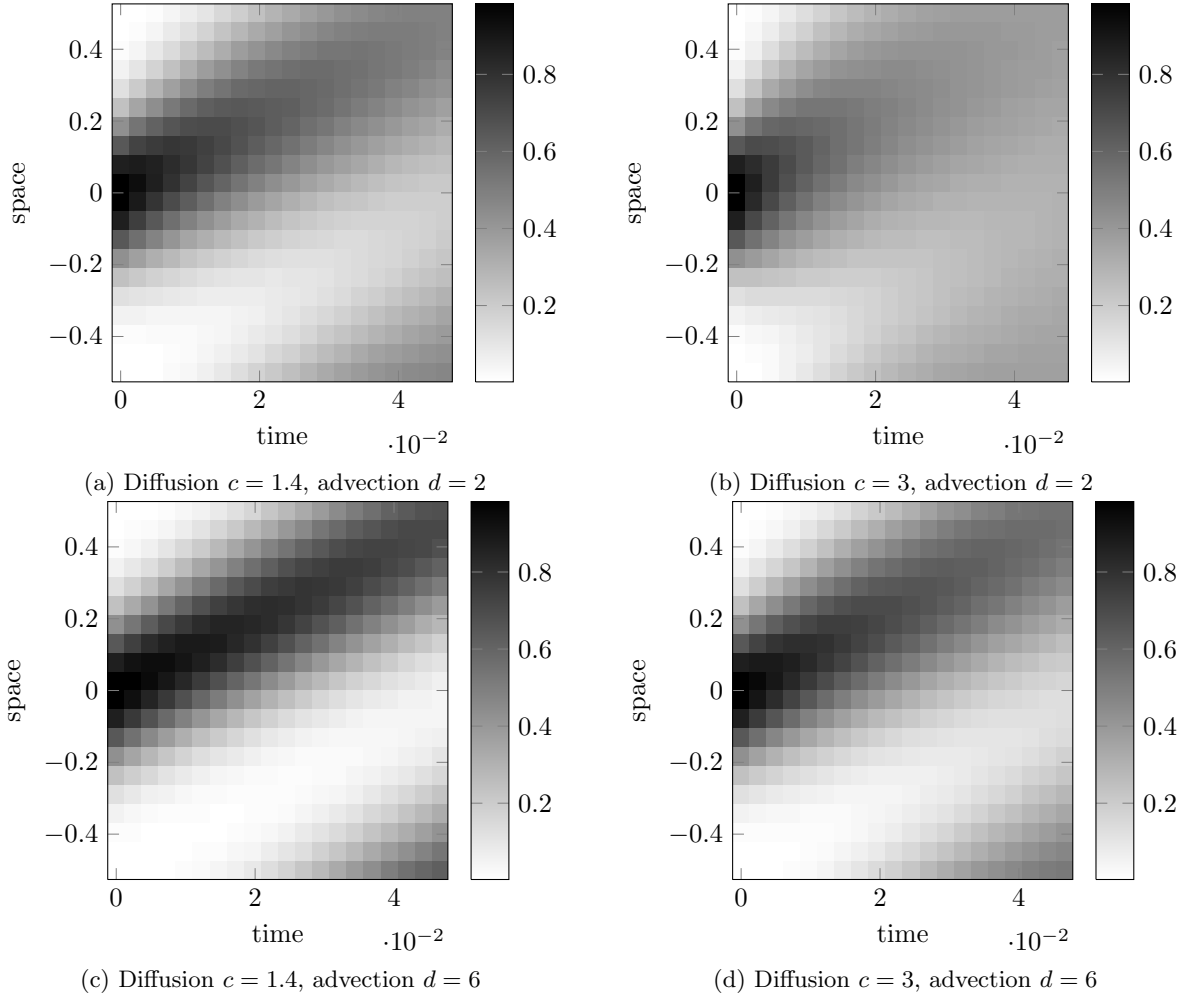


Figure 17: Results for different values of diffusion and advection constants in the PDE example §4.2. Time is oriented horizontally, space vertically.

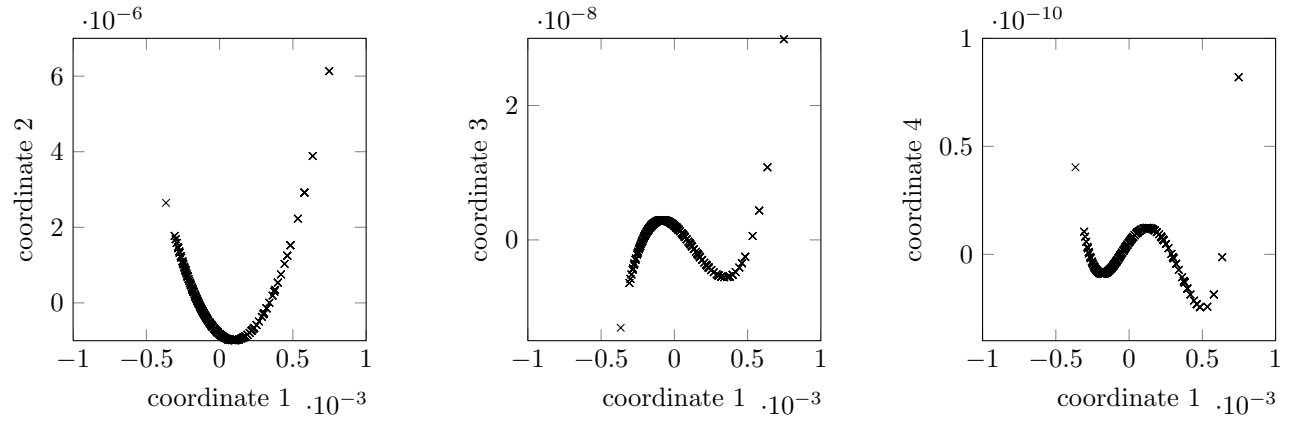


Figure 18: Reduced system coordinates relative to the first. The second, third and fourth coordinates can be expressed using the first, so they do not contribute any additional information to the system.

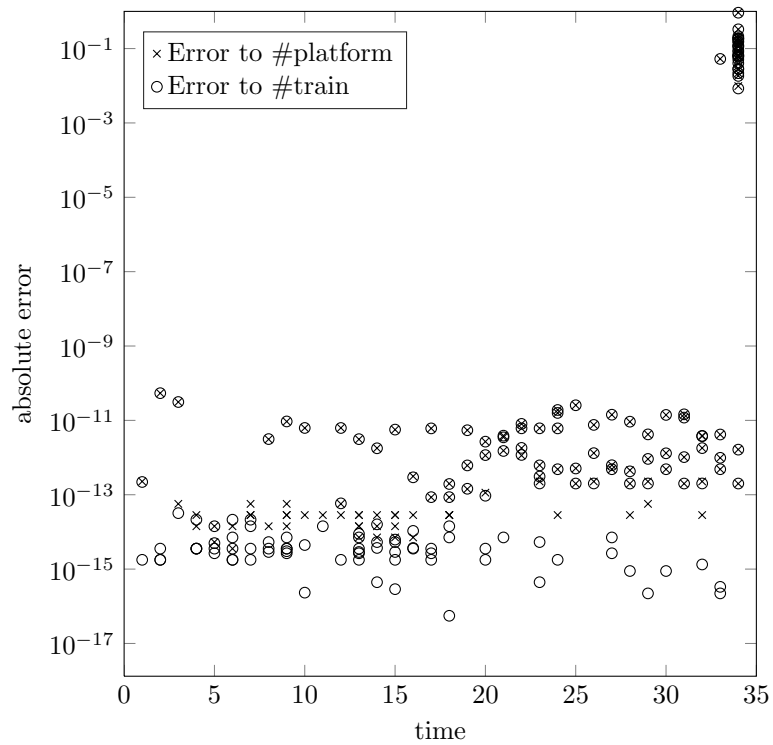


Figure 19: Error between the predictions of the numerical model and the observations used to construct it. For all but the last time steps (where interpolation is no longer valid), the error is at the order of machine precision.

C Acknowledgments

The authors would like to thank Tobias Neckel for fruitful discussions about multiscale modeling. Support from the TopMath Graduate Center of TUM Graduate School at Technische Universität München, Germany is gratefully acknowledged.

References

- [1] JOHN W. BANDLER, QINGSHA S. CHENG, SAMEH A. DAKROURY, AHMED S. MOHAMED, MOHAMED H. BAKR, KAJ MADSEN, AND JACOB SØNDERGAARD, *Space mapping: The state of the art*, IEEE Trans. Microwave Theory Techn., 52 (2004), pp. 337–361.
- [2] MAXIME BARRAULT, YVON MADAY, NGOC CUONG NGUYEN, AND ANTHONY T. PATERA, *An ‘empirical interpolation’ method: application to efficient reduced-basis discretization of partial differential equations*, Comptes Rendus Mathematique, 339 (2004), pp. 667–672.
- [3] T. BERRY, J. R. CRESSMAN, Z. GREGURIĆ-FERENČEK, AND T. SAUER, *Time-scale separation from diffusion-mapped delay coordinates*, SIAM J. Appl. Dyn. Syst., 12 (2013), pp. 618–649.
- [4] A. BRANDT, J. BRANNICK, K. KAHL, AND I. LIVSHITS, *Bootstrap AMG*, SIAM Journal on Scientific Computing, 33 (2011), pp. 612–632.
- [5] HANS-JOACHIM BUNGARTZ AND MICHAEL GRIEBEL, *Sparse grids*, Acta Numerica, 13 (2004), pp. 147–269.
- [6] R. R. COIFMAN, I. G. KEVREKIDIS, S. LAFON, M. MAGGIONI, AND B. NADLER, *Diffusion maps, reduction coordinates, and low dimensional representation of stochastic systems*, Multiscale Modeling & Simulation, 7 (2008), pp. 842–864.
- [7] RONALD R. COIFMAN AND STÉPHANE LAFON, *Diffusion maps*, Applied and Computational Harmonic Analysis, 21 (2006), pp. 5–30.
- [8] ETHAN R. DEYLE AND GEORGE SUGIHARA, *Generalized theorems for nonlinear state space reconstruction*, PLoS ONE, 6 (2011), p. e18295.
- [9] FELIX DIETRICH AND GERTA KÖSTER, *Gradient navigation model for pedestrian dynamics*, Physical Review E, 89 (2014), p. 062801.
- [10] WEINAN E, *Principles of Multiscale Modeling*, Cambridge University Press, Cambridge, 2011.
- [11] BJORN ENGQUIST AND WEINAN E, *The heterogenous multiscale methods*, Communications in Mathematical Sciences, 1 (2003), pp. 87–132.
- [12] STEWART N. ETHIER AND THOMAS G. KURTZ, *Stochastic Processes and Martingales*, Wiley-Blackwell, 1986.
- [13] NEIL FENICHEL, *Persistence and smoothness of invariant manifolds for flows*, Indiana. Univ. Math. J., 21 (1972), pp. 193 – 226.
- [14] DROR GIVON, RAZ KUPFERMAN, AND ANDREW STUART, *Extracting macroscopic dynamics: model problems and algorithms*, Nonlinearity, 17 (2004), pp. 55–127.
- [15] J P HUKÉ AND D S BROOMHEAD, *Embedding theorems for non-uniformly sampled dynamical systems*, Nonlinearity, 20 (2007), pp. 2205–2244.
- [16] IOANNIS G. KEVREKIDIS AND GIOVANNI SAMAËY, *Equation-free multiscale computation: Algorithms and applications*, Annual Review of Physical Chemistry, 60 (2009), pp. 321–344.
- [17] CHAD LIEBERMAN, KAREN WILLCOX, AND OMAR GHATTAS, *Parameter and state model reduction for large-scale statistical inverse problems*, SIAM Journal on Scientific Computing, 32 (2010), pp. 2523–2542.
- [18] EVAN MONROIG, KAZUYUKI AIHARA, AND YOZO FUJINO, *Modeling dynamics from only output data*, Physical Review E, 79 (2009), p. 056208.

- [19] B. MOORE, *Principal component analysis in linear systems: Controllability, observability, and model reduction*, IEEE Trans. Automat. Contr., 26 (1981), pp. 17–32.
- [20] BENJAMIN PEHERSTORFER, DANIEL BUTNARU, KAREN WILLCOX, AND HANS-JOACHIM BUNGARTZ, *Localized discrete empirical interpolation method*, SIAM Journal on Scientific Computing, 36 (2014), pp. A168–A192.
- [21] DAVID RUELLE AND FLORIS TAKENS, *On the nature of turbulence*, Commun.Math. Phys., 20 (1971), pp. 167–192.
- [22] T. SAUER, *Time Series Prediction: Forecasting The Future And Understanding The Past*, Addison-Wesley, Harlow, UK, 1994, ch. Time series prediction by using delay coordinate embedding, pp. 175–193.
- [23] AMIT SINGER, RADEK ERBAN, IOANNIS G. KEVREKIDIS, AND RONALD R. COIFMAN, *Detecting intrinsic slow variables in stochastic dynamical systems by anisotropic diffusion maps*, Proceedings of the National Academy of Sciences, 106 (2009), pp. 16090–16095.
- [24] SIROD SIRISUP, GEORGE EM KARNIADAKIS, DONGBIN XIU, AND IOANNIS G. KEVREKIDIS, *Equation-free/galerkin-free pod-assisted computation of incompressible flows*, Journal of Computational Physics, 207 (2005), pp. 568–587.
- [25] MICHAEL SMALL AND C.K. TSE, *Optimal embedding parameters: a modelling paradigm*, Physica D: Nonlinear Phenomena, 194 (2004), pp. 283–296.
- [26] J. STARK, D.S. BROOMHEAD, M.E. DAVIES, AND J. HUKÉ, *Takens embedding theorems for forced and stochastic systems*, Nonlinear Analysis: Theory, Methods & Applications, 30 (1997), pp. 5303–5314.
- [27] FLORIS TAKENS, *Detecting strange attractors in turbulence*, Lecture Notes in Mathematics, (1981), pp. 366–381.
- [28] ULRICH WEIDMANN, *Transporttechnik der Fussgänger*, vol. 90 of Schriftenreihe des IVT, Institut für Verkehrsplanung, Transporttechnik, Strassen- und Eisenbahnbau (IVT) ETH, Zürich, 2 ed., 1992.